Session 404

# OpenGL
# High Performance 2D

Geoff Stahl
Graphics and Games DTS Engineer

# What You'll Learn

- Why OpenGL for 2D?
- Implementing 2D with OpenGL
  - Image Display
  - Sprites
  - Side Scrolling
  - QuickTime Integration

# Why OpenGL?

- Fastest graphics pipeline
  - Up to 650MB per second from AGP memory
- Flexible API
  - Free 2D/2.5D Orthographic projection with effects
- Hardware Acceleration
  - Most advanced hardware support
- Lots of sample code

# Why Not OpenGL?

- Supports16-and 32-bit display depth acceleration
  - Not optimal solution without acceleration
    - Might not be good fit for 8 bit special cases
- Not as robust imaging model as Quartz 2D
  - PDF, fonts, lines and shapes
- Large API
  - We are here to help…

# Implementing 2D for OpenGL

- Image Display
  - Texture loading
  - Orthographic display and scaling
- Sprites
  - Alpha blending and alpha test
  - Matrix manipulations
- Side Scrolling
  - Large image texturing and scrolling
- QuickTime Integration

# Image Display

- Texture Loading
  - Use QuickTime Image Compression API
  - Packed Pixels
  - 3 methods
    - Scale to power of 2 on load
    - Segment to power of 2 textures
    - **texture_rectangular**
      - Future

# Texture Loading

- QuickTime Image Compression API

```
// build buffer for image
GetGraphicsImporterForFile (&fsspecImage, &giComp);
err = GraphicsImportGetNaturalBounds (giComp, &rectImage);
rowStride = (rectImage.right - rectImage.left) * wTargetDepth / 8;
pBaseAddr = (unsigned char *) NewPtrClear (rowStride *
                                (rectImage.bottom - rectImage.top));
if (wTargetDepth == 32)
        QTNewGWorldFromPtr (&pGWImage, k32ARGBPixelFormat,
                        &rectImage, NULL, NULL, 0, pBaseAddr, rowStride);
else
        QTNewGWorldFromPtr (&pGWImage, k16BE555PixelFormat,
                        &rectImage, NULL, NULL, 0, pBaseAddr, rowStride);
if (NULL == pGWImage)
        return;
```

# Texture Loading

- QuickTime Image Compression API

```
// Set scale matrix for texture
SetIdentityMatrix (&matrix);
ScaleMatrix (&matrix, X2Fix ((float) textureWidth / (float) imageWidth),
                      X2Fix ((float) textureHeight / (float) imageHeight),
                      X2Fix (0.0), X2Fix (0.0));
err = GraphicsImportSetMatrix(giComp, &matrix);
```

# Texture Loading

- QuickTime Image Compression API

```
// scale decompress to gworld
err = GraphicsImportSetGWorld (giComp, pGWImage, NULL);
err = GraphicsImportSetQuality (giComp, codecLosslessQuality);
hPixMap = GetGWorldPixMap (pGWImage);
if ((hPixMap) && (LockPixels (hPixMap))) // lock off screen pixel map
        GraphicsImportDraw (giComp);
UnlockPixels (hPixMap);
CloseComponent (giComp);
```

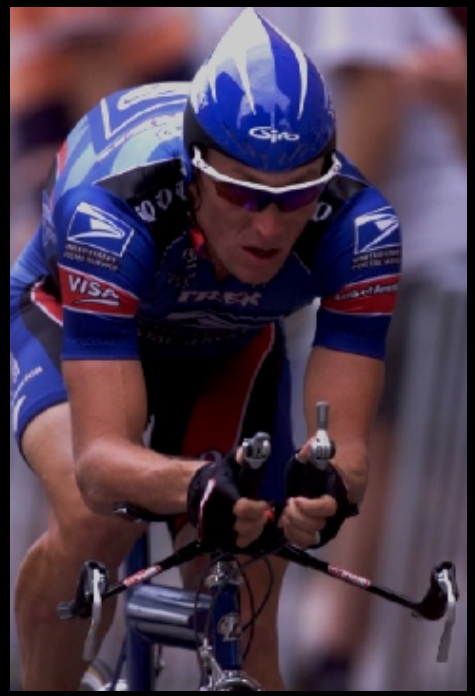
# Texture Loading

- Packed Pixels
  - **AGL_NO_RECOVERY**
  - OpenGL version 1.2 or **GL_APPLE_packed_pixel** extension

```
if (imageDepth == 32)
        glTexImage2D (GL_TEXTURE_2D, 0, GL_RGBA,
                textureWidth, textureHeight, 0,
                GL_BGRA_EXT, GL_UNSIGNED_INT_8_8_8_8_REV,
                pImageBuffer);
else
        glTexImage2D (GL_TEXTURE_2D, 0, GL_RGBA,
                textureWidth, textureHeight, 0,
                GL_BGRA_EXT, GL_UNSIGNED_SHORT_1_5_5_5_REV,
                pImageBuffer);
```

# Texture Loading

- Scale to Power of 2
  - Modifies image aspect ratio
  - Restored when drawn



# Texture Loading

- Scale to Power of 2

```
textureSize = GetTextureDimFromImageDim  (imageSize);

long GetTextureDimFromImageDim (long imageSize) {
        long i = 0, texDim = 1, imageTemp = imageDimension;
        while (imageTemp)
                { imageTemp = imageTemp >> 1; i++; }
        texDim = texDim << i;
        if (texDim > gMaxTextureSize) // if we are too big
                return gMaxTextureSize;
        else if ((texDim - imageDimension) <
                    (imageDimension - (texDim >> 1)))
                return texDim;
        else
                return (texDim >> 1);
}
```

# Demo

## Scaled Texture Image Display

# Texture Loading

- Segment to power of 2
  - Base address
    - Offset to upper left of texture segment
  - Row stride
    - Use image width
    - Set with **UNPACK_ROW_LENGTH**

# Texture Loading

- Segment to power of 2

```
maxTextureSize = gMaxTextureSize;

textureX = GetTextureNumFromTextureDim (textureWidth, maxTextureSize);
textureY = GetTextureNumFromTextureDim (textureHeight, maxTextureSize);

pTextureName = (GLuint *) NewPtrClear (sizeof (GLuint) * textureX * textureY);

glPixelStorei (GL_UNPACK_ROW_LENGTH, textureWidth);
glGenTextures (textureX * textureY, pTextureName);
```

# Texture Loading

- Segment to power of 2 (Cont.)

```
long x, y, k = 0, offsetY = 0, offsetX = 0, currWidth, currHeight;
for (x = 0; x < textureX; x++)
{
        // use remaining to determine next texture size
        currWidth = GetNextTextureSize (textureWidth - offsetX,
                                                maxTextureSize);
        offsetY = 0;
        for (y = 0; y < textureY; y++)
        {
                // buffer pointer is at base + rows * row size + columns
                unsigned char * pBuffer = pImageBuffer +
                                (offsetY * textureWidth * (imageDepth >> 3)) +
                                  offsetX * (imageDepth >> 3);
```

# Texture Loading

- Segment to power of 2 (Cont.)

```
// use remaining to determine next texture size
currHeight = GetNextTextureSize (textureHeight - offsetY,
                                 maxTextureSize);
glBindTexture (GL_TEXTURE_2D, pTextureName[k++]);
if (imageDepth == 32)
        glTexImage2D (GL_TEXTURE_2D, 0, GL_RGBA,
                      currWidth, currHeight, 0, GL_BGRA_EXT,
                      GL_UNSIGNED_INT_8_8_8_8_REV, pBuffer);
else
        glTexImage2D (GL_TEXTURE_2D, 0, GL_RGBA,
                      currWidth, currHeight, 0, GL_BGRA_EXT,
                      GL_UNSIGNED_SHORT_1_5_5_5_REV, pBuffer);
offsetY += currHeight;
}
offsetX += currWidth;
}
```

# Texture Loading

- Segment to power of 2 (Cont.)

```
long GetTextureNumFromTextureDim (long textureDim, short maxTex)
{
        long i = 0;
        if (textureDim > maxTex)
                i = textureDim / maxTex;
         textureDim -= i * maxTex;
        if (textureDim & 0x2000) i++; // 8192
        if (textureDim & 0x1000) i++; // 4096
        if (textureDim & 0x800) i++; // 2048
        if (textureDim & 0x400) i++; // 1024
        if (textureDim & 0x200) i++; // 512
        if (textureDim & 0x100) i++; // 256
        …
        return i;
}
```

# Texture Loading

- Segment to power of 2 (Cont.)

```
long GetNextTextureSize (long textureDimension, short maxTex)
{
        if (textureDim > maxTex)
                return maxTex;
        if (textureDim & 0x2000) return 0x2000; // 8192
         if (textureDim & 0x1000) return 0x1000; // 4096
        if (textureDim & 0x800) return 0x800; // 2048
        if (textureDim & 0x400) return 0x400; // 1024
        if (textureDim & 0x200) return 0x200; // 512
        if (textureDim & 0x100) return 0x100; // 256
        ...
        return 0;
}
```

# Image Display

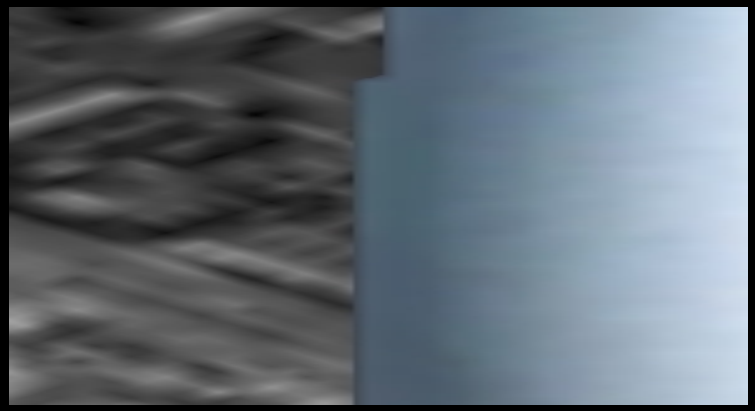
- Orthographic Display
  - Parallel projection
  - Ignore Z
  - Use polygon to scale
    - Could use texture coordinates
  - Use ModelView matrix to offset/rotate

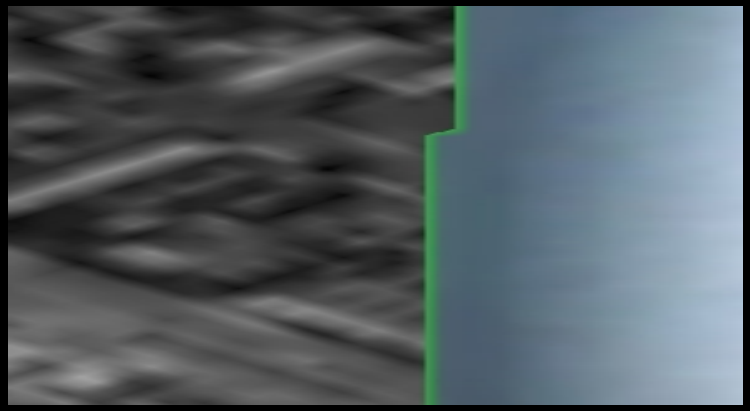
# Demo

## Segmented Texture Image Display

# Sprites

- Sub-texturing
  - Sprite frames stored in texture
    - Use texture coordinates to locate sprite image
    - Watch filtering





# Sprites

- Alpha Masking and Source Color Keying
  - Set alpha on image load
    - Background color pixels
      - alpha = 0
      - pixel = alpha and neutral background color
    - All other pixels
      - pixel |= 0x8000 or 0xFF000000
      - Ensures alpha is set

# Sprites

- Alpha Masking and Source Color Keying

```
if (imageDepth == 32)
{
      unsigned long i;
      unsigned long * pixel = (unsigned long *) pImageBuffer;
      for (i = 0; i < textureWidth * textureHeight; i++)
      {
        // mask only color bits and look for green
        if ((*pixel & 0x00FFFFFF) == 0x00FF00)
                *pixel = 0x00303040; // replace green with dark gray transparent
         else
            *pixel |= 0xFF000000; // ensure alpha is set for opaque pixels
         pixel++;
      }
  }
```

# Sprites

- Alpha Masking and Source Color Keying (Cont.)

```
else // 16 bit
{
        unsigned short i;
        unsigned short * pixel = (unsigned short *) pImageBuffer;
        for (i = 0; i < textureWidth * textureHeight; i++)
        {       // mask only color bits and look for green
                if ((*pixel & 0x7FFF) == 0x03E0)
                        *pixel = 0x0363; // replace green with dark gray transparent
                else
                        *pixel |= 0x8000; // ensure alpha is set for opaque pixels
                pixel++;
        }
}
```

# Sprites

- Alpha blending and transparency
  - Set alpha on image load
    - Use separate alpha image or luminance

```
unsigned long i;
unsigned long * pixel = (unsigned long *) pImageBuffer;
for (i = 0; i < textureWidth * textureHeight; i++)
{
        *pixel &= 0x00FFFFFF;
        *pixel |= (((*pixel & 0x000000FF) + ((*pixel & 0x0000FF00) >> 8) +
                    ((*pixel & 0x00FF0000) >> 16)) / 3) << 24;
}
```

# Sprites

- Drawing with alpha masks and alpha blends

```
// use alpha test to decal sprites
glAlphaFunc (GL_GREATER, 0.5);
if (fAlphaTest)
        glEnable (GL_ALPHA_TEST);
else
        glDisable (GL_ALPHA_TEST);

 // use blend for transparency
 glBlendFunc (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
 if (fAlphaBlend)
        glEnable (GL_BLEND);
 else
        glDisable (GL_BLEND);

glEnable (GL_DEPTH_TEST);
```

# Side Scrolling

- Designed to replicate large parallax scrolled backgrounds
- Use tiled textures to form a large background image
- Use grid of polygons for background
  - Draw grid of polygons
    - Offset by world center
    - Rotated by world rotation
    - Scaled by world scale

# Side Scrolling

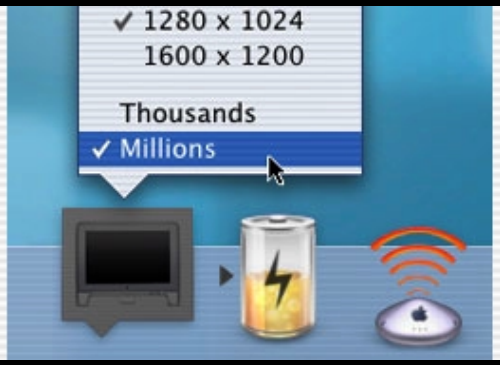
- Texture Handling
  - Load as one image
  - Tile as with segmented images
  - Use **GenTextures** to generate texture name list
  - Let OpenGL load and unload textures as needed
  - Same as segmented images

# Side Scrolling

- Drawing
  - Setup orthographic context
  - Pin world center to sprite of interest
    - Translate others by negative world coords
  - Draw sprites
    - Can grossly cull to visible sprites
    - Let OpenGL do precision culling
    - Use depth buffer for layering

# Demo

## Red Rocket



# QuickTime Integration

- Texture Allocation
  - **QTNewGWorldFromPtr**
    - Allows QT drawing to off screen with no padding
- Texture format
  - Packed pixels for direct texture handling
- QT Synchronization
  - Minimizes texture upload
- Drawing
  - **SubTexImage2D** updates only specified texture data

# QuickTime Integration

- Texture Allocation

```
rowStride = QTTextureWidth * OffScreenDepth / 8;
pBaseAddr = (unsigned char *) NewPtrClear (rowStride * QTTextureHeight);
if (offScreenDepth == 32)
        QTNewGWorldFromPtr (&pOffscreen, k32ARGBPixelFormat,
                &rectMovie, NULL, NULL, 0, pBaseAddr , rowStride );
else
        QTNewGWorldFromPtr  (&pOffscreen, k16BE555PixelFormat,
                &rectMovie, NULL, NULL, 0, pBaseAddr, rowStride);
SetGWorld (pOffscreen, NULL);
SetMovieGWorld (gMovie, (CGrafPtr)pOffscreen, NULL);
```

# QuickTime Integration

- Texture Format (Packed pixels)

```
if (offScreenDepth == 32)
        glTexImage2D (GL_TEXTURE_2D, 0, GL_RGBA,
                      textureSize, textureSize, 0, GL_BGRA_EXT,
else
        glTexImage2D (GL_TEXTURE_2D, 0, GL_RGBA,
                      textureSize, textureSize, 0, GL_BGRA_EXT,
                      GL_UNSIGNED_SHORT_1_5_5_5_REV,
                      pTexture);
```

# QuickTime Integration

- QT Synchronization

```
gMovieDrawUPP = NewMovieDrawingCompleteUPP (MovieDrawProc);
SetMovieDrawingCompleteProc (gMovie, movieDrawingCallWhenChanged,
                                    gMovieDrawUPP, &gfMovieTexUpdate);

...

pascal OSErr MovieDrawProc (Movie movie, long refcon)
{
        #pragma unused (movie)
         *((Boolean *) refcon) = true;
        return noErr;
}
```

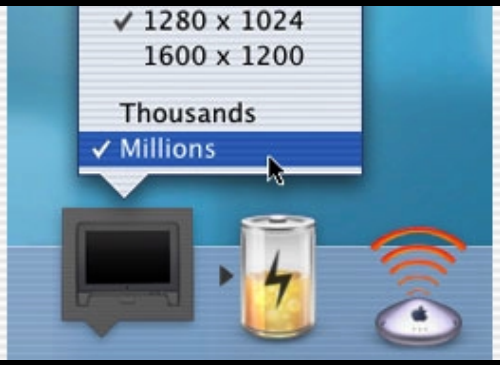
# QuickTime Integration

- Drawing

```
if (gfMovieTexUpdate)
{
        if (offScreenDepth == 32)
                glTexSubImage2D (GL_TEXTURE_2D, 0, 0, 0, textureWidth, textureHeight,
                    GL_BGRA_EXT, GL_UNSIGNED_INT_8_8_8_8_REV, pBaseAddr);
        else
                glTexSubImage2D (GL_TEXTURE_2D, 0, 0, 0, textureWidth, textureHeight,
                    GL_BGRA_EXT, GL_UNSIGNED_SHORT_1_5_5_5_REV, pBaseAddr);
        gfMovieDraw = false;
}
MoveFrame ();
glBegin (GL_QUADS);
        glTexCoord3f (0.0, 0.0, 0.0); glColor3f (1.0, 1.0, 1.0, 1.0); glVertex3f (-1, -1, 0);
        glTexCoord3f (0.0, 1.0, 0.0); glColor3f (1.0, 1.0, 1.0, 1.0); glVertex3f (-1, 1, 0);
        glTexCoord3f (1.0, 1.0, 0.0); glColor3f (1.0, 1.0, 1.0, 1.0); glVertex3f (1, 1, 0);
        glTexCoord3f (1.0, 0.0, 0.0); glColor3f (1.0, 1.0, 1.0, 1.0); glVertex3f (1, -1, 0);
glEnd ();
```

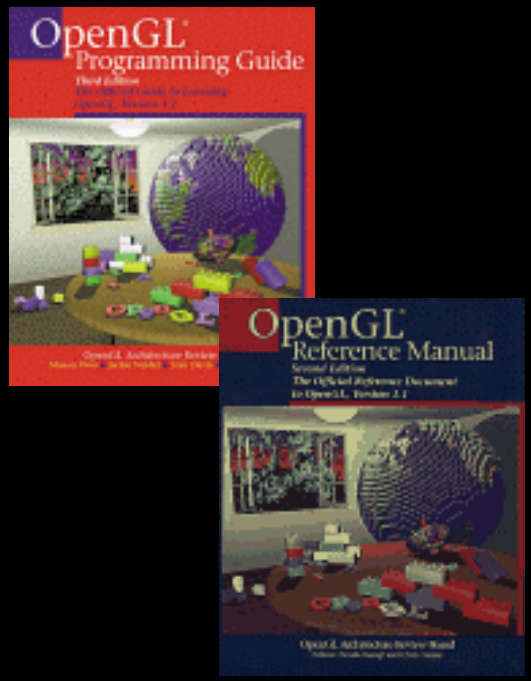
# Demo

## QuickTime Integration



# References

- In print
  - OpenGL Programming Guide
    - The red book
  - OpenGL Reference Manual
    - The blue book
- Online
  - **<http://www.opengl.org>**
  - **<http://lists.apple.com>**
    - Search for “opengl”
  - **<http://developer.apple.com/opengl>**



# Wrap Up

- OpenGL is a powerful 2D/2.5D API
  - Provides robust and flexible tool set
  - Allows robust hardware acceleration
  - New Paradigm to solve real world problems
- Samples
  - **<http://developer.apple.com/samplecode/>**

# Roadmap

| | |
|---|---|
| **405 OpenGL: Geometry and Modeling** | Room J2<br>**Wed., 5:00pm** |
| **408 OpenGL: Optimization** | Room J2<br>**Thurs., 9:00am** |
| **409 OpenGL: Advanced Rendering** | Room J2<br>**Thurs., 10:30am** |
| **FF110 Feedback Forum: OpenGL** | Room J1<br>**Thurs., 5:00pm** |
| **FF109 Feedback Forum: QT** | Room J1<br>**Thurs., 3:30pm** |

# Who to Contact

**Sergio Mello**
**3D Graphics Technology Manager**
Apple Worldwide Developer Relations
**sergio@apple.com**

# Q&A

http://developer.apple.com/events/urls.html

WWDC
Worldwide Developers Conference 2001